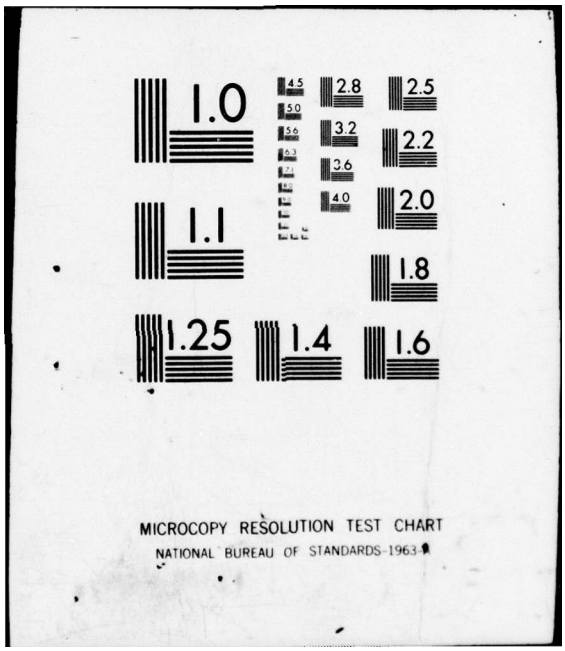
AD-A052 729 CHARLES STARK DRAPER LAB INC CAMBRIDGE MA F/G 9/2
JOVIAL STRUCTURED DESIGN DIAGRAMMER (JSDD). VOLUME I. REPORT SU--ETC(U)
FEB 78 G GODDARD, M WHITWORTH , E STROVINK F30602-76-C-0408
UNCLASSIFIED R-1120-VOL-1 RADC-TR-78-9-VOL-1 NL

1 OF 1
AD
A052 729

END
DATE
FILMED
5-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

RADC-TR-78-9, Vol I (of four)
Final Technical Report
February 1978

JOVIAL STRUCTURED DESIGN DIAGRAMMER (JSDD), Volume I. Report Summary.

G. Goddard
M. Whitworth
E. Strovink

The Charles Stark Draper Laboratory, Inc.

Approved for public release; distribution unlimited.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-78-9, Vol I (of two) has been reviewed and is approved for publication.

APPROVED: Donald VanAlstine

DONALD VANALSTINE
Project Engineer

APPROVED: Marshall C. Bannister

MARSHALL C. BANNISTER, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISIM) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

(18) RADC (19) TR-78-9-VOL-1

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-78-9, Vol I (of four)
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

(6) Volume I.

4. TITLE (and Subtitle): JOVIAL STRUCTURED DESIGN DIAGRAMMER (JSDD). Report Summary.

(9) 5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. Sep 76 - Oct 77.

(14) 6. PERFORMING ORG. REPORT NUMBER: R-1128-VOL-1

(10) 7. AUTHOR(s): G. Goddard, M. Whitworth, E. Strovink

(15) 8. CONTRACT OR GRANT NUMBER(s): F30602-76-C-0408

9. PERFORMING ORGANIZATION NAME AND ADDRESS: The Charles Stark Draper Laboratory, Inc., 555 Technology Square, Cambridge MA 02139

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: P.E. 62702F, J.O. 55811412

11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISIM), Griffiss AFB NY 13441

(11) 12. REPORT DATE: Feb 1978

13. NUMBER OF PAGES: 8 (12) 12P.

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same (16) 5581 (17) 14

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

DDC APR 17 1978 F

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Donald VanAlstine (ISIM)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Structured programming	Preprocessor
Structured design diagram	Flowcharter
Structured extension	JOVIAL J3
Parse	Invocation diagram
Parser generator	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report summarizes the implementation of the prototype JOVIAL structured Design Diagrammer (JSDD).

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409 386 JOB

EVALUATION

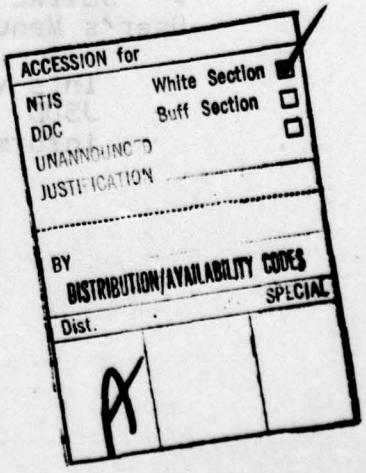
The objective of this effort was to implement a system to automate the documentation of structured JOVIAL source statements. This system, the JSDD, functions as an aid to the programmer/ design engineer in producing software systems written in accordance with structured programming conventions. This development falls within the goals of RADC TPO V, specifically in the 3.3 Tools and Procedures area.

The Charles Stark Draper Laboratory, Inc. implemented the JSDD on the Honeywell 6180 Computer System at RADC using the GCOS III Encapsulator operating under the MULTICS remote user system. The JSDD is designed to be transportable to batch, as well as remote batch GCOS operating environments. It was written using structured JOVIAL syntax and contains a built-in JOVIAL pre-compiler for code translation for the existing target JOVIAL J3 compiler currently available on the H6180. As a test case, the JSDD modules were used as source data for the system and appeared to generate correct code when passed on to the JOVIAL J3 compiler.

The JSDD provides a graphic representation of the sometimes complex logical sequences that constitute the structure of largescale software development efforts. It is for this reason that the Design Diagrammer fills an important place in the spectrum of software tools currently being developed under the RADC advanced software program.

Donald L. Van Alstine

DONALD L. VANALSTINE
Project Engineer



REPORT SUMMARY

This document was produced to satisfy the requirements of contract number F30602-76-C-0408 with the Rome Air Development Center. It is one of four companion volumes:

* JOVIAL Structured Design Diagrammer (JSDD) Report Summary

  This document is a summary of the contents of the JSDD Final Report.

* JOVIAL Structured Design Diagrammer (JSDD) Final Report

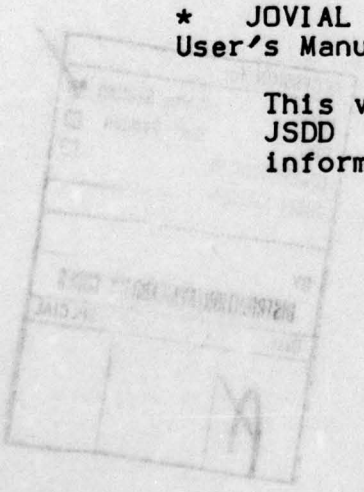
  This volume presents the design techniques for implementing the JSDD and describes the use of Structured Design Diagrams.

* JOVIAL Structured Design Diagrammer (JSDD) Program Description

  This volume presents a detailed description of the program implementation for purposes of maintaining and/or modifying the JSDD.

* JOVIAL Structured Design Diagrammer (JSDD) User's Manual

  This volume presents the user's view of the JSDD along with user options and other information about running the program.



Acknowledgement

This report was prepared by The Charles Stark Draper Laboratory, Inc., under Contract F30602-76-C-0408 with the Rome Air Development Center at Griffis Air Force Base.

Especial credit is due Margaret Hamilton, who pioneered principles of Structured Programming at Draper Laboratory. Saydean Zeldin originally suggested the symbology implemented in the output of the JOVIAL Structured Design Diagrammer. Thanks should go also to William Daly, who created the Structured Design Diagrammer for the HAL language (currently being used on the NASA Space Shuttle project). The authors are indebted to Victor Voydock for his invaluable assistance in implementing a complete MULTICS user interface which was used successfully for the duration of the JSDD implementation. The authors are also grateful to J. Barton DeWolf whose many suggestions were of great assistance throughout this effort.

1. Introduction

In recent years, the digital computer software industry has directed considerable effort toward the development of design and implementation methodologies to ensure the sufficiency, reliability, and maintainability of software systems. The most widely known product of this effort is the loosely defined set of design and programming practices called "Structured Programming."

Structured Programming does not constitute a complete software development methodology. Rather, it is a collection of general guidelines for use by software designers and implementors. As such, it provides no uniform approach to system design and offers no method of evaluating system sufficiency with respect to requirements or design. Despite these shortcomings, adherence to Structured Programming principles can be of great assistance in producing software systems which are reliable and intellectually manageable.

The techniques of Structured Programming are sufficiently general to allow system developers a tremendous amount of stylistic freedom. However, the generality of the techniques has made the development of a standard approach to software analysis extremely difficult. The prototype JOVIAL Structured Design Diagrammer (JSDD) is the first component of an integrated software analysis and documentation system which will address itself to this task.

The JSDD processes digital computer programs written in either JOVIAL J3 or Extended JOVIAL J3. Extended JOVIAL J3 is standard JOVIAL J3 with the addition of structured extensions.

The JSDD is implemented in JOVIAL J3 as a three program system that is designed to run on a Honeywell Information Systems, Inc., Series 6000 computer supporting GCOS Version 1/G. The implementation work was conducted on the Rome Air Development Center's MULTICS computing facility via the ARPA Network. Development work was performed on the GCOS Encapsulator which is available under the MULTICS operating system. The MULTICS environment provided most of the software tools that were employed during the implementation.

The JSDD is an automated analysis and documentation system which produces two types of diagrams: Structured Design

Diagrams (SDDs) and Invocation Diagrams.

2. Structured Design Diagram Description

Structured Design Diagrams (SDDs) provide a graphic two dimensional display of the nested logical sequences that define the structure of a computer program.

SDDs for JOVIAL J3 are constructed from two basic structural elements: pentagonal and rectangular boxes. The rectangular box is used to contain JOVIAL statements that are executed in sequence. Pentagonal boxes are used to contain JOVIAL constructs which modify other JOVIAL constructs (e.g., an If or For Clause).

3. Invocation Diagram Description

Invocation Diagrams are a display of a software system's functional (calling) structure. The Invocation Diagrammer produces two different outputs: (1) a list of procedures that are members of one or more recursive invocation loops, and (2) the Invocation Diagram itself.

The first output, if it appears on the diagram, occurs before the actual diagram under the heading "ULTIMATELY SELF-RECURSIVE." Under it are listed all procedures that call themselves, either directly or indirectly. An example of a direct recursive call is a procedure which contains, as part of its code, a call to itself. Indirectly recursive calls are best illustrated, again, by an example. Suppose procedure A can call procedure B which can call procedure C. If, as part of its code, procedure C contains a call to procedure A, all three procedures (A, B, and C) can theoretically call themselves.

4. Design of the JOVIAL Structured Design Diagrammer

The JSDD has two conceptual passes. "Pass 1" (consisting of the Design Diagram Database Generator) performs the tasks of analyzing the syntax of an input program and creating a data base for use by the second pass. "Pass 2" (consisting of the Design Diagram Generator and the Invocation Diagrammer) uses the data base created by Pass 1 to construct Structured Design Diagrams (SDDs) and Invocation Diagrams. A two pass design is motivated by two factors. First, it is desirable to separate language dependent functions from language independent functions. Such a separation facilitates the

adaptation of Pass 2 to target languages other than JOVIAL J3. Second, the two pass design provides a great deal of flexibility in the formatting of diagrams. Pass 2 of the JSDD can produce diagrams having a wide variety of formats from the data base produced by a single Pass 1 execution.

In parsing input source programs, Pass 1 acts as a table-driven deterministic pushdown automaton (DPDA). The tables which drive the Pass 1 parse are the product of an LALR(k) parser generator that accepts a syntactic description of a language as input and outputs parsing tables for the language.

The two Pass 2 programs (the Design Diagram Generator (DDG) and Invocation Diagrammer) interpret the Pass 1 generated data base, and create diagrams in accordance with the formatting specifications in the DDG options compool (see User's Manual). The DDG is implemented as a two part program. First it maps out the Structured Design Diagram and creates a temporary data base containing the mapping information. Only then does it produce the actual diagram. This strategy allows it to calculate forward and backward referencing information (in case a diagram overflows the page width) and a Table of Contents, without committing prematurely to any hard-copy output. If adjustments need to be made in the diagram, the temporary data base can be modified easily.

5. Defining the JOVIAL J3 Syntax

The parsing tables used by Pass 1 are generated by an LALR(k) parser generator. The parser generator accepts the syntactic description of a language (in BACKUS-NAUR form) and produces the parsing tables for the language.

The parser generator can produce tables for any LALR(k) grammar where k is finite.

There is no grammar processable by a left to right parser generator (having an finite upper bound on required lookahead) which will produce tables which will parse all JOVIAL J3 programs and which will parse no inputs which are not JOVIAL J3 programs.

In order to sidestep this problem, the JOVIAL J3 grammar has been "expanded" so that all JOVIAL J3 programs can be parsed. However, as a result of this expansion, there are

sentences which the grammar can generate which are not valid JOVIAL J3 programs.

The syntax of JOVIAL J3 is relatively complex for a programming language. This complexity requires that JOVIAL's BNF description contain a very large number of productions. In attempting to generate parsing tables, it was found that the size of JOVIAL's BNF description exceeded an internal limit imposed upon the parser generator's input grammars. In order to avoid a costly investigation of the parser generator's limits, work on the JSDD was based on an early, slightly inaccurate version of the JOVIAL grammar. The Design Diagrammer Data Base Generator has been adjusted to enable it to successfully parse the full set of valid JOVIAL programs.

6. The Structured Extensions to JOVIAL J3

Since JOVIAL J3 lacks certain structured programming mechanisms which eliminate the need for Goto statements, these mechanisms have been added as allowable programming features in programs submitted to the JSDD. It is assumed that such programs would have to be subjected to a preprocessor before submission to a JOVIAL J3 compiler. Such a preprocessor has been supplied as a deliverable item with the JSDD.

The current JOVIAL J3 structured extensions are:

1) The DO WHILE LOOP
2) The DO UNTIL LOOP
3) The CASE STATEMENT

Programs incorporating structured extensions are translated into standard JOVIAL J3 programs by the JOVIAL Extended Structures Translator (JEST preprocessor). JEST is a PL/I program implemented on the RADC MULTICS system.

7. Conclusions and Recommendations

From the implementor's point of view the JOVIAL J3 language was not a good choice of a programming language in which to implement the JSDD. For instance, the JOVIAL J3 compiler supplied for use on this contract was incapable of optimizing the JSDD programs. Even if this were not the case, the JSDD would still run more slowly than necessary, because much computer time is consumed performing operations for which the JOVIAL J3 compiler is not very efficient. For

example, a major shortcoming of the compiler is that it does not support random access output operations on disk files. This induces the JSDD to consume great amounts of computer time doing double buffered I/O, and requires the inclusion of additional software modules in the JSDD computer programs.

Two additional aspects of the JOVIAL J3 language render it less than desirable for the JSDD application or for implementing compiler-like tools in general. First, the static nature of JOVIAL's data handling makes it difficult to do dynamic memory management. Second, JOVIAL does not contain string handling constructs that are naturally suited to compiler-like programming. The outstanding difficulty with the JOVIAL character string manipulation capability is that the current string length and the "declared" maximum string length for a string variable are not available at execution time. In order to circumvent the string handling difficulties, the character string handling package is incorporated into each JSDD program. This package offers string operations such as substring and concatenation.

Despite the difficulties attendant to the implementation, the JSDD produces Structured Design Diagrams of high quality. The JOVIAL programmer should find these diagrams to be of great assistance during program design and implementation as well as being useful for documentation purposes. The Invocation Diagrammer produces an output that is extremely useful and probably should long have been a standard feature of commercial compilers. The prototype JSDD has been designed such that it can also serve as the nucleus of a comprehensive automated documentation system for JOVIAL programs, should the construction of such a system be desired at a later time.

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.